October 2009

Uploading User–Defined Functions onto the AMIDAS Website

Chung-Lin Shan

School of Physics and Astronomy, Seoul National University
Seoul 151-747, Republic of Korea
E-mail: cshan@hep1.snu.ac.kr

Abstract

The AMIDAS website has been established as an online interactive tool for running simulations and analyzing data in direct Dark Matter detection experiments. At the first phase of the website building, only some commonly used WIMP velocity distribution functions and elastic nuclear form factors have been involved in the AMIDAS code. In order to let the options for velocity distribution as well as for nuclear form factors be more flexible, we have extended the AMIDAS code to be able to include *user–uploaded* files with their own functions. In this article, I describe the preparation of files of user–defined functions onto the AMIDAS website. Some examples will also be given.

1 Introduction

In the last few years we developed new methods for analyzing data, i.e., measured recoil energies, from (future) direct Dark Matter detection experiments as model–independently as possible [1, 2, 3, 4]. These methods will help us to understand the nature of WIMP (Weakly Interacting Massive Particle χ) Dark Matter, to identify them among new particles produced hopefully in the near future at colliders, as well as to reconstruct the (sub)structure of our Galactic halo. Following the development of these model–independent data analysis procedures, we combined the programs for simulations to a compact system: `AMIDAS` (A Model–Independent Data Analysis System). For users' convenience and under the collaboration with the ILIAS Project [5], an online system has also been established at the same time [6, 7].

For the first version of the `AMIDAS` code and website, the options for target nuclei, for the velocity distribution function of halo WIMPs, as well as for the elastic nuclear form factors for spin–independent (SI) and spin–dependent (SD) WIMP–nucleus interactions are fixed and only some commonly used forms have been involved in the `AMIDAS` code [7]. Users can not choose different detector materials nor use different velocity distribution/form factors for their simulations and/or data analyses. In order to let the options for the velocity distribution as well as for the nuclear form factors be more flexible, we have extended the `AMIDAS` code to be able to include *user–uploaded* files with their own functions. Note that, since the `AMIDAS` code has been written in the C programming language, all user–defined functions for uploading must also be given using the syntax of C.

The remainder of this article is organized as follows. In Sec. 2 I will talk about setting users' own target nuclei. In Secs. 3 and 4 the preparation of files defining the velocity distribution function and the nuclear form factors will be described, respectively. I will conclude in Sec. 5. Some intrinsically defined constants and functions in the `AMIDAS` code will be given in an appendix.

2 Target nuclei

Users can give as usual the element symbol and the atomic mass number A of their chosen target nuclei on the website directly. For the determination of ratios of different WIMP–nucleon couplings/cross sections, the total spin of the target nucleus J and the expectation values of the proton and neutron group spins $\langle S_{(\mathrm{p,n})} \rangle$ for the chosen targets with spin sensitivities are also required.

3 Velocity distribution of halo WIMPs

For defining the one–dimensional WIMP velocity distribution, users should in practice give the integral over the velocity distribution function:

$$\int_{v_{\mathrm{min}}=\alpha\sqrt{Q}}^{v_{\mathrm{max}}} \left[\frac{f_1(v,t)}{v} \right] dv \tag{1}$$

with the name of "`Intf1v_v_user`". Note here that the lower limit of the integral, the minimal incoming velocity of incident WIMPs that can deposit the energy Q in the detector, v_{min}, must be expressed as a function of the energy Q through $v_{\mathrm{min}} = \alpha\sqrt{Q}$; the upper limit of the integral, v_{max}, is often set as the escape velocity v_{esc} or even ∞, since the WIMP flux on the Earth is

usually assumed to be negligible at velocities $v \geq v_{\rm esc} \gtrsim 600$ km/s. Meanwhile, since

$$\alpha \equiv \sqrt{\frac{m_{\rm N}}{2m_{\rm r,N}^2}} \tag{2}$$

with the reduced mass

$$m_{\rm r,N} \equiv \frac{m_\chi m_{\rm N}}{m_\chi + m_{\rm N}} \tag{3}$$

is a function of the WIMP mass m_χ and the mass of target nucleus $m_{\rm N}$, which has been defined as a function of the atomic mass number A, the expression of the integral over the WIMP velocity distribution should also be a function of m_χ and A. Finally, the velocity distribution function and thus the expression of the integral should generally be a function of time t.

As examples, the integral over the simple Maxwellian velocity distribution function [8, 1] can be defined as

Example

```
double Intf1v_v_user(double mchi, int A, double QQ, double tt)
{
  return
    (2.0 / sqrt(M_PI) / (v_0 * v_U)) *
    exp(-alpha(mchi, A) * alpha(mchi, A) * QQ /
         (  (v_0 * v_U) * (v_0 * v_U)  )       );
}
```

and the integral over the shifted Maxwellian velocity distribution [8, 1] can be given by

Example

```
double Intf1v_v_user(double mchi, int A, double QQ, double tt)
{
  return
    (1.0 / 2.0 / (v_e(tt) * v_U)) *
    (  erf( (alpha(mchi, A) * sqrt(QQ) + (v_e(tt) * v_U)) / (v_0 * v_U) )
     - erf( (alpha(mchi, A) * sqrt(QQ) - (v_e(tt) * v_U)) / (v_0 * v_U) ) );
}
```

Here `v_e(tt)`, standing for the *time–dependent* Earth's velocity in the Galactic frame $v_{\rm e}(t)$, has been defined in the `AMIDAS` code by

Definition

```
double v_e(double tt)
{
  return v_0 * ( 1.05 + 0.07 * cos(omega * (tt - t_p)) );
}
```

Note that, firstly, `v_U`, the function `alpha(mchi, A)`, where `mchi` and `A` stand for the WIMP mass m_χ and the atomic mass number of the target nucleus, A, and the constant `omega` are defined intrinsically in the `AMIDAS` code. Secondly, `v_0` and `t_p`, standing for the Sun's orbital velocity in the Galactic frame, v_0, and the date on which the velocity of the Earth relative to the WIMP halo is maximal, $t_{\rm p}$, are two input parameters which users can set later on the website [7]. Thirdly, `QQ` and `tt` stand for the energy variable Q and time variable t. Finally,

for a *time–independent* velocity distribution or a numerical expression with a fixed experimental running time $t = t_{\rm expt}$, "`double tt`" should still be declared as one of the *four* variables of the function `Intf1v_v_user`.

On the other hand, for comparing results of the reconstructed velocity distribution with the (input) theoretical one in the output plots, users need an *extra file* for giving the velocity distribution function *itself* (not the integral over it now) for drawing the (input) theoretical velocity distribution function $f_1(v)$. Since the `gnuplot` software [9] has been used for drawing output plots, the velocity distribution function given in this file must be written using the syntax of `gnuplot` with the name of "`f1v_user(x)`". As examples, the simple Maxwellian velocity distribution can be given as

Example

```
M_PI = 3.1416

v_0 = 220.0

   f1v_user(x)                      \
=  (4.0 / sqrt(M_PI)) *             \
   ((x * x) / (v_0 * v_0 * v_0)) *  \
   exp(-(x * x) / (v_0 * v_0))      \
```

and for the shifted Maxwellian velocity distribution:

Example

```
M_PI = 3.1416
omega  = 2.0 * M_PI / 365.0

t_p    = 152.5
t_expt = 243.75

v_0 = 220.0
v_e = v_0 * ( 1.05 + 0.07 * cos(omega * (t_expt - t_p)) )

  f1v_user(x)                                          \
=  1.0 / sqrt(M_PI) * (x / v_e / v_0) *                \
   (  exp(-(x - v_e) * (x - v_e) / (v_0 * v_0))        \
    - exp(-(x + v_e) * (x + v_e) / (v_0 * v_0))  )     \
```

Note here that the "\" (backslash) must be used (also at the end of the last line!) in order to insert the definition given in this file into the other intrinsic commands correctly.

Two sample files, one is for the `AMIDAS` code and the other one is for the `gnuplot` software, can be downloaded from the `AMIDAS` website.

4 Nuclear form factors

For defining users' own nuclear form factors, *not only* the definition of the *squared* form factor $F^2(Q)$ *but also* its derivative with respect to the energy Q:

$$\frac{dF^2(Q)}{dQ} = 2F(Q)\left[\frac{dF(Q)}{dQ}\right] \tag{4}$$

(not $F(Q)$ itself nor $dF(Q)/dQ$) must be given in *one* file together with the names of "`FQ_SI_user` (`FQ_SD_user`)" and "`dFQdQ_SI_user` (`dFQdQ_SD_user`)". The expression of the squared form factor can be either a general analytic form for different nuclei or a specified form for the chosen nucleus. As examples, for the SI WIMP–nucleus cross section, the exponential form factor $F^2_{\rm ex}(Q)$ [10, 11, 8] can be given as

Example

```
double FQ_SI_user(int A, double QQ)
{
  return exp(-QQ / Q_0(A));
}

double dFQdQ_SI_user(int A, double QQ)
{
  return -(1.0 / Q_0(A)) * FQ_SI_user(A, QQ);
}
```

and the Woods–Saxon form factor $F^2_{\rm WS}(Q)$ [12, 8]:

Example

```
double FQ_SI_user(int A, double QQ)
{
  if (QQ == 0.0)
    return 1.0;

  else
  {
    return
      ( 3.0 * sphBesselj(1, qq(A, QQ) * R_1(A)) / (qq(A, QQ) * R_1(A)) ) *
      ( 3.0 * sphBesselj(1, qq(A, QQ) * R_1(A)) / (qq(A, QQ) * R_1(A)) ) *
      exp(-(qq(A, QQ) * ss) * (qq(A, QQ) * ss));
  }
}

double dFQdQ_SI_user(int A, double QQ)
{
  if (QQ == 0.0)
    return -0.4 * (R_A(A) * R_A(A)) * m_N(A);

  else
  {
    return
      (  dsphBesselj(1, qq(A, QQ) * R_1(A)) * R_1(A) /
          sphBesselj(1, qq(A, QQ) * R_1(A))
       - 1.0 / qq(A, QQ)
       - (qq(A, QQ) * ss * ss) ) *
      FQ_SI_user(A, QQ) * ((2.0 * m_N(A)) / qq(A, QQ));
  }
}
```

Here the spherical Bessel functions

$$j_n(x) = \sqrt{\frac{2}{\pi x}}\, J_{n+1/2}(x)\,, \tag{5}$$

their derivatives

$$j'_n(x) = \sqrt{\frac{2}{\pi x}}\left[-\frac{1}{2x}\, J_{n+1/2}(x) + J'_{n+1/2}(x)\right]\,, \tag{6}$$

as well as the (derivatives of the) half–integer Bessel functions $J_{n+1/2}(x)$, for $n = 0,\ \pm1,\ \pm2,\ \cdots$, are given in an intrinsic package of the `AMIDAS` code. The functions `Q_0(A)`, `qq(A, QQ)`, `R_1(A)`, `R_A(A)`, `m_N(A)`, and the constant `ss` are also defined intrinsically in the `AMIDAS` code (see the appendix).

As a more complicated example, the thin–shell form factor $F^2_{\rm TS}(Q)$ used sometimes for the SD WIMP–nucleus cross section [13, 14] can be defined as

Example

```
double FQ_SD_user(int A, double QQ)
{
  if (QQ == 0.0)
    return 1.0;

  else
  if (QQ <= QQ_SD_min(A) ||
      QQ >= QQ_SD_max(A)   )
  {
    return
      sphBesselj(0, qq(A, QQ) * R_1(A)) * sphBesselj(0, qq(A, QQ) * R_1(A));
  }

  else
    return FQ_TS_const;
}

double dFQdQ_SD_user(int A, double QQ)
{
  if (QQ == 0.0)
    return -(1.0 / 3.0) * (R_1(A) * R_1(A)) * (2.0 * m_N(A));

  else
  if (QQ <= QQ_SD_min(A) ||
      QQ >= QQ_SD_max(A)   )
  {
    return
      dsphBesselj(0, qq(A, QQ) * R_1(A)) * R_1(A) *
       sphBesselj(0, qq(A, QQ) * R_1(A)) * ((2.0 * m_N(A)) / qq(A, QQ));
  }

  else
    return 0.0;
}
```

The functions `QQ_SD_min(A)`, `QQ_SD_max(A)`, and the constant `FQ_TS_const` are defined in the `AMIDAS` code (see the appendix).

5 Summary

In this article, I described the preparation of files giving user–defined WIMP velocity distribution function and/or elastic nuclear form factor(s) which can be uploaded onto the `AMIDAS` website for more flexible simulations or data analyses. This improvement allows theorists to simulate with their own/favorite models and compare them with (future) experimental results, as well as gives experimentalists flexible choices for more suitable form factor(s) for their own detector materials.

In summary, up to now all basic functions of the `AMIDAS` code and website have been well established. Hopefully this new tool can help our colleagues to detect/discover WIMP Dark Matter, to understand the nature of Dark Matter particles and the (sub)structure of the Galactic halo in the future.

Acknowledgments

The author would like to thank James Y. Y. Liu for discussing the basic idea of combining user–defined functions with the source code. The author would be grateful to the ILIAS Project and the Physikalisches Institut der Universität Tübingen for kindly providing the opportunity of the collaboration and the technical support of the `AMIDAS` website. The author would also like to thank the friendly hospitality of the Institute of Physics, National Chiao Tung University where part of this work was completed. This work was partially supported by the BK21 Frontier Physics Research Division under project no. BA06A1102 of Korea Research Foundation.

A Relevant constants and functions defined in the `AMIDAS` code

Here I list some relevant constants and functions defined intrinsically in the `AMIDAS` code, which could be needed by users for defining their own functions.

A.1 Defined constants

The constants defined in the `AMIDAS` code are in *natural units* as following:

Definitions

```
double c = 1.0;
double v_U = c / (2.998 * 1e5);
double m_U = 1e6 / (c * c);
double fm_U = c / (0.1973 * 1e6);
double m_p = 0.938 * m_U;
double omega = 2.0 * M_PI / 365.0;
```

A.2 Defined functions

The functions used for defining the commonly used WIMP velocity distribution functions and nuclear form factors in the `AMIDAS` code are given here.

A.2.1 The reduced mass $m_{\rm r,N}(m_\chi, A)$ and $\alpha(m_\chi, A)$

All masses in the `AMIDAS` code are in the unit of GeV/c^2. Firstly, the mass of the target nucleus is defined as a function of the atomic mass number A by

Definition

```
double m_N(int A)
{
  return m_p * A * 0.99;
}
```

Here the mass difference between a proton and a neutron has been neglected. And the reduced mass between the WIMP mass and "something" is given by

Definition

```
double mchi_r(double mchi, double mx)
{
  return mchi * mx / (mchi + mx);
}
```

Therefore, the reduced mass of the WIMP mass with the mass of the target nucleus, $m_{\rm r,N}$ in Eq.(3), and with the proton mass are defined as

Definitions

```
double mchi_rN(double mchi, int A)
{
  return mchi_r(mchi * m_U, m_N(A));
}

double mchi_rp(double mchi)
{
  return mchi_r(mchi * m_U, m_p);
}
```

Finally, α defined in Eq.(2) has been given as a function of m_χ and A:

Definition

```
double alpha(double mchi, int A)
{
  return sqrt(m_N(A) / 2.0) / mchi_rN(mchi, A);
}
```

A.2.2 For nuclear form factors

For the exponential form factor $F_{\rm ex}(Q)$, the nuclear radius R_0 and the nuclear coherence energy Q_0 have been defined as functions of the atomic mass number A:

Definitions

```
double R_0(int A)
{
  return (0.3 + 0.91 * cbrt(m_N(A) / m_U)) * fm_U;
}
```

```
double Q_0(int A)
{
  return 1.5 / (m_N(A) * R_0(A) * R_0(A));
}
```

For the Woods–Saxon form factor $F_{\rm WS}(Q)$, the nuclear skin thickness s, the effective nuclear radius R_A, and the radius R_1 are defined as

Definitions

```
double ss = fm_U;

double R_A(int A)
{
  return 1.2 * cbrt(A) * fm_U;
}

double R_1(int A)
{
  return sqrt(R_A(A) * R_A(A) - 5.0 * ss * ss);
}
```

Meanwhile, the transferred 3–momentum $q = \sqrt{2m_{\rm N}Q}$ has been give as a function of the atomic mass number A and recoil energy Q:

Definition

```
double qq(int A, double QQ)
{
  return sqrt(2.0 * m_N(A) * QQ);
}
```

For the thin–shell form factor $F_{\rm TS}(Q)$, the dimensionless constants giving the lower and upper energy bounds, between which the form factor is a constant, and this constant ($\simeq 0.047$) are given as

Definitions

```
double qqR_1_min = 2.55;
double qqR_1_max = 4.50;

double FQ_TS_const;
FQ_TS_const = sphBesselj(0, qqR_1_min) * sphBesselj(0, qqR_1_min);
```

Then the lower and upper energy bounds can be estimated by

Definitions

```
double QQ_SD_min(int A)
{
  return (qqR_1_min * qqR_1_min) / (2.0 * m_N(A) * R_1(A) * R_1(A));
}

double QQ_SD_max(int A)
{
  return (qqR_1_max * qqR_1_max) / (2.0 * m_N(A) * R_1(A) * R_1(A));
}
```

A.3 Declared variables

Here I list the *short used names* for variables and constants in the `AMIDAS` code. Note that these names should be *avoided* to use in the user–uploaded code(s), otherwise the `AMIDAS` program might not work correctly.

Definitions

```
AX, AX_str, Ax, calN, fp_U, G_F, hn, InX, JJ, k_1, ka, kn, lambda, ln,
mchi_assumed, mchi_tmp, mm, m_N_TX, nexpt, nmchi, NmX, nn, nn_max, nn_win,
nplot, nranap, nTX, pb_U, pm, Qmax, Qmid, Qmin, Qwin, Qvin, Qtp, rho, rho_0,
rho_U, RJ, RJX, rlh, Rn, Rn_sh, RnX, Rsigma, R_TX, run, runi, runj, runk,
r_win, sh, Sn, Snp, sol, Sp, Spn, std_rho_0, std_t_p, std_v_0, s_U, t_end,
t_expt, tmax, tmid, tmin, t_start, TX, v_e_annual, v_esc
```

References

[1] M. Drees and C. L. Shan, *J. Cosmol. Astropart. Phys.* **0706**, 011 (2007).

[2] M. Drees and C. L. Shan, *J. Cosmol. Astropart. Phys.* **0806**, 012 (2008).

[3] M. Drees and C. L. Shan, *proceedings of IDM 2008*, `arXiv:0809.2441 [hep-ph]` (2008).

[4] M. Drees and C. L. Shan, *proceedings of DARK 2009*, `arXiv:0903.3300 [hep-ph]` (2009).

[5] `http://www-ilias.cea.fr/`.

[6] `http://pisrv0.pit.physik.uni-tuebingen.de/darkmatter/amidas/`.

[7] C. L. Shan, *proceedings of SUSY09*, `arXiv:0909.1459 [astro-ph.IM]` (2009).

[8] G. Jungman, M. Kamionkowski, and K. Griest, *Phys. Rep.* **267**, 195 (1996).

[9] `http://www.gnuplot.info/`.

[10] S. P. Ahlen *et al.*, *Phys. Lett.* **B 195**, 603 (1987).

[11] K. Freese, J. Frieman, and A. Gould, *Phys. Rev.* **D 37**, 3388 (1988).

[12] J. Engel, *Phys. Lett.* **B 264**, 114 (1991).

[13] J. D. Lewin and P. F. Smith, *Astropart. Phys.* **6**, 87 (1996).

[14] H. V. Klapdor-Kleingrothaus, I. V. Krivosheina, and C. Tomei, *Phys. Lett.* **B 609**, 226 (2005).